(2)

Technical Report 1443
September 1991

Modeling the Density of a Distribution Containing a Jump Nonstationarity

D. J. Marchette

ADMINISTRATIVE INFORMATION

The funding for this work was provided by Office of Chief of Naval Research, Independent Research Programs (IR), under the program element 0601152N.

Released by
D. L. Conwell, Head
Research Development
and Architectures Branch

Under authority of
J. A. Salzmann, Head
Ashore Command
and Intelligence Centers
Division

JT

EXECUTIVE SUMMARY

OBJECTIVE

Construct a method for nonparametric density estimation in the presence of a nonstationary distribution. Evaluate the performance of the method.

RESULTS

1. An algorithm using the adaptive mixtures algorithm for nonparametric density estimation was constructed. Modifications to the basic algorithm allow it to adapt its model to changing distributions.

2. The algorithm was tested in the special case of jump nonstationarity. Its performance was compared to a windowed kernel estimator approach to the same problem.

RECOMMENDATIONS

1. Establish performance criteria for use in general nonstationary problems.

2. Investigate the performance of the windowed kernel estimator with adaptive bandwidth.

3. Modify the adaptive mixtures algorithm to use a change detector, rather than a windowed approach. This would allow the algorithm to get better estimates when the density is stationary, and still allow it to track the nonstationary distributions.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

INTRODUCTION

The problem of estimating a probability density function occurs in many areas of statistics, pattern recognition, and control. In many cases, the densities that occur in real problems are nonstationary, so a technique for estimating these densities would be extremely useful.

In this work, we focus on a simple type of nonstationarity, which we refer to as a "jump nonstationarity." We assume we have independent measurements taken from a univariate random variable X that is stationary and distributed as d_1 before time t_J, and stationary and distributed as d_2 after time t_J. We say a jump occurs at time t_J if $d_1 \neq d_2$.

In many problems, we assumed the densities are known and the goal is to detect the jump as soon after it occurs as possible. Often, however, the densities are not known, and one must estimate them using one of the many techniques for density estimation. We do not assume that the densities are known, or that the time of the jump is known. Also, rather than focusing on the detection of the jump, we focus on the estimation of the densities. Since we wish to make no parametric assumptions about the densities, we require a nonparametric technique.

An application of this work can be envisioned by considering a sensor that is monitoring some process, where the probability density function of the process is of interest. This process arises in a number of monitoring, detection, and classification problems. Assume the sensor is not under the control of the data analyst and that it is subject to "upgrades" without warning, these upgrades effecting the distribution of the measurements made by the sensor. Thus, the data have a potential jump at each upgrade time.

In addition to the requirement that the densities be modeled, we further put a processing restraint on the estimator. Due to high data rates in many problems, we require our estimator to be recursive. That is, the estimator must update its model using only the currently available measurement, rather than retaining a number of measurements and iterating on these data to refine the estimate. Thus we can think of the data as coming to us one point at a time, and we are required to report our estimate of the density with each new data point.

This work is related to the work of Carlstein (1988), but with an important difference: Carlstein is interested in detecting a change-point, or jump, while we are trying to model the density. However, the flavor of the two approaches is very similar. Both use the idea of windowing the data and estimating the distribution within the window. Carlstein's method is to break the data into two contiguous windows and estimate the distribution functions of the data within the two windows. If these distribution functions are different, then a jump has occurred at the point between the two windows.

A recursive version of this could be envisioned where the two windows are of fixed size and move across the data in time, stopping when a change has been detected. Since we are interested in only the density of the data, we need only retain one window, although the approach given could easily be used for a change-point detector by estimating the density twice, with a lag between the two estimators.

The estimator described in this work is the Adaptive Mixture (AM) estimator. This estimator will be described in detail, but the idea is to model the density as a mixture of Gaussian (Normal) distributions. The number of components of the mixture is variable, and the data are windowed to allow the estimator to quickly adapt to a change in the distribution. We will compare this estimator to a common nonparametric estimator: the kernel estimator.

WINDOWED KERNEL (WK) ESTIMATOR

While the histogram is perhaps the simplest and most widely used nonparametric estimator, we will consider a related estimator, the kernel estimator, and its windowed version, the windowed kernal (WK) estimator. This WK estimator takes the N most recent points, where N is the window width, and computes a kernel estimator for the density. We will assume a Gaussian kernel, and a fixed bandwidth h The estimator is then

$$\hat{f}(x) = \sum_{i=1}^{N} \frac{1}{\sqrt{2\pi}\, Nh} \mathrm{Exp}\left[-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2\right]. \tag{1}$$

Modifications can be made to this to allow the h to be chosen from the data, or to have a different h for each component in the sum, but these will not be pursued here.

The purpose of considering the WK estimator is to give a benchmark against which the performance of the AM can be measured. It can be viewed as a recursive technique, since the new estimate can be formed from the old estimate by replacing the oldest data point with the newest. It is also easy to analyze the performance of the kernel estimator, eliminating the need for time consuming simulations.

ADAPTIVE MIXTURES (AM)

One nonparametric technique for the estimation of densities, similar to the kernel estimator, is to model the density as a (finite) mixture of a given distribution. The parameters of the mixture can be estimated iteratively using the EM method or recursively using a recursive version of the EM method (Titterington et al., 1985). Assuming the number of components and initial parameters of the components are chosen appropriately, one can obtain a good approximation to a wide range of densities. Throughout, we will assume the mixture is a mixture of Gaussian distributions.

The AM model is a modification of the techniques of mixture models to allow the number of components to grow with the number of data points. The addition of a component to the mixture requires the specification of the mixture proportion, the mean, and the variance. The algorithm is given below, with a pseudocode implementation, given in the appendix. The idea is to give the new component a proportion related to the number of components, a fixed initial variance, and a mean corresponding to the data point x that initiated the addition. If our algorithm for deciding to add a component was to always add, and we let the variance of all the components decrease with the number of points, we would have a Gaussian kernel estimator.

By allowing the number of components to change with time, we have made the parametric mixture estimator into a nonparametric estimator. The border between parametric and nonparametric is rather ill-defined, so some may argue with this. Although some might prefer to call this approach semiparametric, we prefer to think of it as a nonparametric technique. It is important to note the algorithm does not assume that the number of components of the density is known, or that it is even a mixture, although the estimator is only applicable if the density is well approximated by a mixture.

The formula for an n-component univariate Gaussian mixture is

$$\hat{f}(x) = \sum_{i=1}^{n} \frac{P_i}{\sqrt{2\pi\, \mathrm{var}_i}} \mathrm{Exp}\left[-\frac{1}{2}\left(\frac{x - \mathrm{mean}_i}{\mathrm{var}_i}\right)^2 \right]. \tag{2}$$

We call p_i the mixture proportion, and obviously we require the p_i's to be positive and sum to 1.

We consider two ways to determine when to add a component. The simplest is to add a component after a fixed number of points have been processed. Thus, we start with one component, estimate its parameters for a fixed time T, then add a new component to the mixture, estimate the new parameters for T points, etc. This approach has the advantage of being simple to implement and tractable to analyze, but has the disadvantage that the number of components grows linearly in the number of data points. Also, the time T must be chosen large enough to allow the parameters to adapt, but small enough to allow the flexibility needed for a wide range of problems.

Another approach is to let the data determine the number of components. The idea is to add a component whenever the new point x is in a region of low density. This approach allows the system to quickly detect the appearance of a new mode. The algorithm is given in the appendix in pseudocode, but we will describe it here for completeness. Each component is a Gaussian density, and we denote the i^{th} component density, with mean μ_i and variance v_i, evaluated at the point x by $N(x;\mu_i, v_i)$. Let

$$s_i(x) = \frac{N(x; \mu_i, v_i)}{N(\mu_i; \mu_i, v_i)} \tag{3}$$

$$s(x) = \underset{i}{\mathrm{Max}}(s_i(x)). \tag{4}$$

Then we add a component if $s(x) < C$ for a fixed (user defined) constant C. We call C the create threshold. Thus we add a component if x is "far" from the mean of any existing component, where "far" is determined by the above equations.

A drawback of this approach is that if the density is not well modeled by a small number of components, say for instance if the density is uniform, we would really like the opposite rule: to add a component whenever the new point is in a region of high density. However, when a jump occurs, the new density has no *a priori* relationship to the old, and so our estimator must be sensitive to new regions of density. We also want the algorithm to use as few components as necessary, and to stop creating when the model is "good enough." The solution taken by this algorithm is to make C large if a large number of components is necessary, or to start with a large number of components if *a priori* information is available. Thus we define "good enough" to be met when the support of the density is adequately covered by the components of the estimate.

If the criterion for the addition of a component is not met, we want to use the new data point to update the current model. This update allows us to improve the estimate, and to adjust the estimate after the jump has occurred. The method used is the recursive version of the proportional update (Titterington, 1984). That is, we update each component proportionally to the (estimated) probability that the datum is from that component. Specifically, we have

$$\mathrm{amt}_i = \frac{P_i \hat{f}_i(x_{n+1})}{\hat{f}(x_{n+1})} \tag{5}$$

$$\mathrm{var}_i = \mathrm{var}_i + \frac{\mathrm{amt}_i}{W + \mathrm{amt}_i - 1}\left(\frac{W}{W + \mathrm{amt}_i}(\mathrm{mean}_i - x_{n+1})^2 - \mathrm{var}_i\right) \tag{6}$$

$$\mathrm{mean}_i = \mathrm{mean}_i + \frac{\mathrm{amt}_i}{W}(x_{n+1} - \mathrm{mean}_i) \tag{7}$$

$$p_i = p_i + \frac{1}{W}(\mathrm{amt}_i - p_i). \tag{8}$$

Here, $\hat{f}_i(x_{n+1}) = N(x_n + 1; \mathrm{mean}_i, \mathrm{var}_i)$ is the value of the ith component at the point x_{n+1}, and W, the window width, is a fixed constant that will be explained. The important point is these formulas use only the current estimate and the most recent data point, and thus are truly recursive.

Although we will be looking exclusively at the univariate case, note these equations can easily be modified for use on multivariate data. The only changes necessary are to view equation 7 as a vector equation, and equation 6 as a matrix equation, where $(\text{mean}_i - x_{n+1})^2$ is to be viewed as the outer product of $(\text{mean}_i - x_{n+1})^2$ with itself (the usual quadratic term in the sample variance calculation).

To allow the algorithm to weight new points more heavily than old points, and thus change to modeling the new distribution after the jump, we window the data. To see how this works, consider the case of a single Gaussian, and the problem of estimating the mean. The recursive formula for the sample mean calculation (Chan et al., 1983) is

$$\bar{x}_{n+1} = \bar{x}_n + \frac{1}{n+1}(x_{n+1} - \bar{x}_n) \tag{9}$$

where x_n is the nth data point, and $\bar{x}_0 = 0$. At time n, this gives the sample mean for the first n data points. We can make this sensitive to changes in the distribution by putting an exponential window on the data, accomplished by choosing a constant W, called the window width, and modifying equation 9 to

$$\bar{x}_{n+1} = \bar{x}_n + \frac{1}{W}(x_{n+1} - \bar{x}_n) = \left(1 - \frac{1}{W}\right)\bar{x}_n + \frac{1}{W}x_{n+1}. \tag{10}$$

Converting this recursion to a summation we have

$$\bar{x}_{n+1} = \left(1 - \frac{1}{W}\right)^n x_1 + \frac{1}{W}\sum_{j=o}^{n-1}\left(1 - \frac{1}{W}\right)^j x_{n-j+1}. \tag{11}$$

Note, the contribution of a point to the estimate decreases as $(1 - \frac{1}{W})^n$. This gives a measure of how fast the algorithm can react to a change in the distribution. Taking expectations and simplifying we see that

$$\begin{aligned} E\bar{x}_{n+1} &= \left\{\left(1 - \frac{1}{W}\right)^n + \frac{1}{W}\sum_{j=o}^{n-1}\left(1 - \frac{1}{W}\right)^j\right\}EX \\ &= \left\{\left(1 - \frac{1}{W}\right)^n + 1 - \left(1 - \frac{1}{W}\right)^n\right\}\mu \qquad (12) \\ &= \mu \end{aligned}$$

where $\mu = EX$, and thus the estimate of the mean is unbiased. Also,

$$\begin{aligned}\operatorname{Var}\bar{x}_{n+1} &= \left\{\left(1-\frac{1}{W}\right)^{2n}+\frac{1}{W^2}\sum_{j=o}^{n-1}\left(1-\frac{1}{W}\right)^{2j}\right\}\operatorname{Var}X \\ &= \left\{\left(1-\frac{1}{W}\right)^{2n}+\frac{1}{W^2}\frac{1-\left(1-\frac{1}{W}\right)^{2n}}{1-\left(1-\frac{1}{W}\right)^{2}}\right\}v \\ &= \left\{\left(1-\frac{1}{W}\right)^{2n}+\frac{1-\left(1-\frac{1}{W}\right)^{2n}}{2W-1}\right\}v \\ &= \left\{\frac{(2W-2)\left(1-\frac{1}{W}\right)^{2n}+1}{2W-1}\right\}v\end{aligned} \tag{13}$$

where $v = \operatorname{Var} X$, and so $\operatorname{Var}\bar{x}_{n+1} \dashrightarrow \frac{1}{2W-1}\operatorname{Var} X$ as $n \dashrightarrow \infty$. Compare this with the varianre of the sample mean, which is $\frac{1}{n+1}\operatorname{Var} X$. This moti\ates the description of the constant W as a window on the data: it is as though we took the most recent 2W–2 points in our estimator.

The recursive version of the sample variance is

$$\begin{aligned}\hat{v}_{n+1} &= \hat{v}_n + \frac{1}{n}\left\{\frac{n}{n+1}(x_{n+1}-\hat{\bar{x}}_n)^2 - v_n\right\} \\ &= \frac{n-1}{n}\hat{v}_n + \frac{1}{n+1}(x_{n+1}-\bar{x}_n)^2.\end{aligned} \tag{14}$$

There are several choices that could be made to window this estimator. We will use

$$\hat{v}_{n+1} = \frac{W-1}{W}\hat{v}_n + \frac{1}{W+1}(x_{n+1}-\bar{x}_n)^2 \tag{15}$$

$$= \frac{1}{W+1}\sum_{i=1}^{n}\left(\frac{W-1}{W}\right)^{n-i}(x_{i+1}-\bar{x}_i)^2. \tag{16}$$

Taking expectations, we see

$$E\hat{v}_{n+1} = \frac{1}{W+1}\sum_{i=1}^{n}\left(\frac{W-1}{W}\right)^{n-i} E(x_{i+1}-\bar{x}_i)^2$$

$$= \frac{1}{W+1}\sum_{j=0}^{n-1}\left(\frac{W-1}{W}\right)^{j}\{\text{Var } X + \text{Var } \bar{x}_{n-j}\}$$

$$= \frac{w}{w+1}\left(1-\left(\frac{w-1}{w}\right)^{n}\right)v + \frac{1}{W+1}\sum_{j=0}^{n-1}\left(\frac{W-1}{W}\right)^{j}\frac{(2W-2)\left(\frac{W-1}{W}\right)^{2(n-j-1)}+1}{2W-1}\,v$$

$$= \left\{\frac{2W^2}{2W^2+W-1}\left(1-\left(\frac{W-1}{W}\right)^{n}\right)+\frac{2W-2}{2W^2+W-1}\sum_{j=0}^{n-1}\left(\frac{W-1}{W}\right)^{2n-j-2}\right\}v \qquad (17)$$

$$= \left\{\frac{2W^2}{2W^2+W-1}\left(1-\left(\frac{W-1}{W}\right)^{n}\right)+\frac{2W-2}{2W^2+W-1}\left(\frac{W-1}{W}\right)^{n-1}\sum_{k=0}^{n-1}\left(\frac{W-1}{W}\right)^{k}\right\}v$$

$$= \left\{\frac{2W^2}{2W^2+W-1}\left(1-\left(\frac{W-1}{W}\right)^{n}\right)+\frac{2W^2-2W}{2W^2+W-1}\left(\frac{W-1}{W}\right)^{n-1}\left(1-\left(\frac{W-1}{W}\right)^{n}\right)\right\}v$$

$$\text{-->}\ \frac{2W^2}{2W^{2+}W-1}v \qquad \text{as } n \text{ -->} \infty \qquad (18)$$

so the variance calculation is slightly asymptotically biased. For the window width $W = 25$, which will be used in the simulations below, we have $E\bar{v}_{n+1}$ --> 0.98 v.

Note, we used the fact that x_{i+1} and $\bar{x}_i$ are independent in order to calculate $E(x_{i+1}-\bar{x}_i)^2 = E(\ (x_{i+1}-\mu) + (\mu-\bar{x}_i)\ )^2 = E(x_{i+1}-\mu)^2 + E(\mu-\bar{x}_i)^2$.

We could make the variance estimator asymptotically unbiased by multiplying equation 15 by $\frac{2W^2+W-1}{2W^2}$, but for the sake of simplicity of the algorithm we will use equation 15.

We can evaluate the performance of the estimator computed using equations 10 and 15 by comparing the density constructed with the true density. This comparison is done by taking the mean integrated squared error (MISE), computed by taking the difference of the estimate with the true density, squaring the result, then taking the expectation with respect to X (recall that the estimate is a function of the random variable X). This expectation gives us a measure of how far from the true density we expect the estimate to be.

We can compute the integrated squared error (ISE) of the estimator for the single component case. This is

$$\begin{aligned}\text{ISE} &= \int_{-\infty}^{\infty} \left\{ \frac{1}{\sqrt{2\pi\,v}} \text{Exp}\left(-\frac{(y-\mu)^2}{2v}\right) - \frac{1}{\sqrt{2\,\pi\,\hat{v}}} \text{Exp}\left(-\frac{(y-\bar{x})^2}{2\hat{v}}\right) \right\}^2 dy \\ &= \frac{1}{\sqrt{2\pi\,v}} + \frac{1}{\sqrt{2\pi\,\hat{v}}} - \frac{2}{\sqrt{2\pi}\sqrt{v+\hat{v}}}\ \text{Exp}\left(-\frac{1}{2}\left\{ \frac{\mu^2}{v} + \frac{\bar{x}^2}{\hat{v}} - \frac{\left(\frac{\mu}{v}+\frac{\bar{x}}{v}\right)^2}{\frac{1}{v}+\frac{1}{\hat{v}}} \right\}\right).\end{aligned} \tag{19}$$

Unfortunately, it is difficult to compute the expectation of this MISE even for the simplest estimators of μ and v. Just to get a feel for the lower bound of this estimator, we make some simplifying assumptions. Assume v is known and consider the problem of just estimating the mean. Further, for simplicity let $\mu = 0$, $v = 1$. Then the ISE becomes

$$\text{ISE} = \frac{1}{\sqrt{\pi}} - \frac{1}{\sqrt{\pi}}\ \text{Exp}\left(-\frac{\bar{x}^2}{4}\right). \tag{20}$$

Under these assumptions, this gives approximately (taking the first two terms of the Taylor series)

$$\begin{aligned}\text{MISE} &= \frac{1}{4\sqrt{\pi}} \text{Var}\ \bar{x} \\ &\dashrightarrow \frac{1}{4\sqrt{\pi}\,(2W-1)}.\end{aligned} \tag{21}$$

In the case of $W = 25$, this gives a MISE of ≈ 0.0029. A simulation was run with $n = 1000$, resulting in a MISE of ≈ 0.0025 (obtained by averaging over 100 runs). Compare this with equation 20 computed for the sample mean in place of the windowed mean estimator. Here we get a value of

$$\text{MISE} = \frac{1}{\sqrt{\pi}\,(2n+1)} \tag{22}$$

after n points. This calculation gives a value of 0.0029 for n = 98 = 4W–2. Combining equations 21 and 22 and solving for n in terms of W, we have

$$n \approx \frac{8W-5}{2}. \tag{23}$$

This gives more credence to our claim that the constant W acts as a window on the data, although it gives a different value for the window size than equation 13, and we have a measure of the best MISE that can be expected in the simplest case.

From the above analysis we can see the effect of imposing the window W on the parameter estimates for the single component case. Even in the single component case we needed to simplify to make explicit calculations. In the general case, the analysis is even more difficult and we will rely on simulations to discover its properties.

One more point needs to be made. In the single component case, we take the initial variance, $\hat{v}_1$, to be 0. In the general case, we must be able to compute the posterior probability estimate, equation 5. Since the number of components is allowed to grow, this involves evaluating each component on the new data point, even if the component has "seen" only a single point. Thus we must initialize the variance of each new component as it is added to the model. The choice of this initial variance can effect the performance of the estimator in much the same way that the choice of bandwidth effects the kernel estimator. A small variance (relative to the underlying density) will cause more components to be added early on, while a large variance will tend to make the estimator model the density with a single component. Thus, like the bandwidth of the kernel estimator, the larger the initial variance the smoother the estimator will tend to be. This initial variance v_1 adds a term of the form $(\frac{W-1}{W})^n v_1$ to the variance in equation 15, and so its effect on a single component decreases with time.

PERFORMANCE ANALYSIS

The analysis of the performance of a density estimate in the case of a nonstationary density is not a simple task. In the case of a jump nonstationarity, however, there are useful measures that can be applied. We will consider two such measures, which will be described: (1) MISE between the estimate and the true distribution on each side of the jump, and (2) the point of "detection" of the jump. First we will describe the experiments performed, and then we will describe the performance measures.

THE EXPERIMENT

The basic experiment is to consider a random variable that is distributed as d_1 for time t = 1,...,500 and d_2 for time t > 500. In the case of simulations, 1000 data points are generated from this distribution. Neither the AM nor the WK know when the jump occurs.

Three different experiments were run. These were

E1: $d1 = N(0,1)$ $d2 = N(3,1)$

E2: $d1 = N(0,1)$ $d2 = N(0,0.1)$

E3: $d1 = \frac{1}{2}N(0,1) + \frac{1}{2}N(1,0.5)$ $d2 = \frac{1}{2}N(3,1) + \frac{1}{4}N(2,0.5) + \frac{1}{4}N(-1,0.5)$

where N(m,v) is the normal distribution with mean m and variance v. While these tests are by no means exhaustive, they do give an indication of the performance of the two approaches and the flexibility of the AM approach. The densities are pictured in figures 1, 9, and 14.

For the AM, 100 simulations were performed for each density. This allows averaging of the performance measures to get an estimate of expected performance. The performance measures for the kernel estimator are computed directly.

The choice of bandwidth for the kernel estimator is a problem. In the case where the two distribution are normals with equal variance, experiment 1, the bandwidth can be chosen to be the optimal for these distributions (it is the same for d_1 and d_2 in this case), but for more general distributions the optimal for d_1 is not equal to the optimal for d_2. In this case we compromise, and take the bandwidth to be the bandwidth that would be optimal for the component of the true distributions with the smallest variance. Experience shows this is a reasonable choice for the densities used in the experiments. Note that in general these optimal bandwidths are unknown, and the choice of bandwidth must be made based solely on samples from d_1, or from any *a priori* knowledge about the two distributions that might be available.

On all three distributions two different AM estimators were run: one with a create threshold C of 0.1, and one where a component was added every 50 data points. In addition, for the first two experiments, where the densities are all Gaussian, an AM was run that consisted of a single Gaussian, and for the last experiment, a three-component AM was run.

Note that these experiments were chosen to show the AM approach in its best light. They are all mixtures of Gaussians, and hence the appropriate density to use the adaptive measures on. Although the AM approach can be used on nonmixture distributions, it performs best on mixture data.

MEAN INTEGRATED SQUARED ERROR (MISE)

MISE is a common measure of performance of density estimates for stationary data (Silverman, 1986), and we use the same measure on the segments in which the signal is stationary. For the AM simulations, we compute the ISE between the true distribution and the estimate after each data point, and then we average over the 100 simulations. The figures below show these graphically. For the WK estimator, the MISE can be computed directly since the true distributions are known.

POINT OF DETECTION (PoD)

Although this work is not concerned with the detection of the jump per se, but rather with the estimation of the densities, one can define the PoD to be the number of points after the jump that the MISE between the estimate and d_1 first equals the MISE between the estimate and d_2. Since d_1 and d_2 are not assumed known, this is of no use in practice, but it does give a useful measure of how fast the estimator is detecting the change in the distributions.

In the case of the AM, the PoD is estimated from simulations while for the WK estimator it is one-half the window width. For the purposes of comparison, the simulations were run and the PoD for each experiment was estimated for the AM. Then the MISE was computed for a kernel estimator with the same PoD. Finally, where kernel estimator's performance was poor, a kernel estimator with approximately the same MISE as the AM is shown, and the PoD of this new kernel estimator is compared with the AM.

RESULTS

The results for the three experiments are shown in the figures. The results are shown for the AM, the version of the AM that creates every 50 points, the AM restricted to a single component in experiments 1 and 2, and the AM restricted to three components for experiment 3. The MISE for the WK is shown as a straight line. MISE is computed assuming all the data comes from a single distribution, and so is not plotted for N less than the kernel width or when the window overlaps the jump. Note, the MISE for the WK varies from figure to figure. This variance is because for each algorithm, the PoD is different, requiring a different window and different bandwidth for the kernel estimator.

To talk about asymptotic results, we will use the word "stabilize" rather than "converge." When we say an estimator has stabilized, we mean its MISE has reached (or become close to) its asymptotic limit. In the case of the WK estimator, this means that the window contains only points from a single distribution. In the case of the AM, we say the estimator has stabilized when the MISE is approximately flat.

When looking at the figures, it is important to keep in mind the position of the jump (always at point 500), the PoD (indicated by an X on the x-axis), and the time the estimator takes to stabilize. This will be discussed for the individual experiments.

Tables 1 through 3 tabulate the MISE for the different algorithms. The MISE tabulated is the average MISE for the last 100 points before the jump, and the last 100 points in the experiment. In the tables, kernel 1 and kernel 2 correspond respectively to the kernel estimator with the same PoD as the AM, and the kernel estimator with approximately the same MISE as the AM estimator directly above them in the table. AM 50 corresponds to the AM estimator that creates every 50 points, Normal corresponds to the AM constrained to a single component, and AM 3 corresponds to the AM forced to create only three components. The #Components column shows the number of Gaussians each estimator used (rounded up to the nearest integer in the case of the AM).

The results for the first two experiments are shown in figures 1 through 13. These experiments involve the estimation of single Gaussian distributions, and so we show the performance of the single component version of the AM in figures 7, 8, and 13. Note, the single component version is the best in that its PoD is smallest, and its MISE is smallest from among the AM estimators. The single component version performs as well as the best kernel estimator, with a smaller PoD and (obviously) fewer components required. This result is as expected, since in this case the AM is a parametric estimator, and so would be expected to have the best performance.

The first experiment is designed to show the performance on a distribution where the mean changes at the jump (figure 1). This is one of the easiest cases, and all the methods perform well. In figure 2 we see that for equal PoD's, the AM has a much smaller MISE than the kernel estimator, but is much slower to stabilize. In figure 3 we see that even with equal MISE, the AM is slower to stabilize than the WK estimator. In this experiment, we would expect the AM to add components immediately after the jump, since the density has shifted its mass, and we see this effect in figure 4. As in all these experiments, the AM that creates every 50 points performs the best, and in figures 5 and 6 we see it outperforms the WK estimator. Recall that the processing requirements of this estimator are less than the kernel, with a total number of components of 20. However, unlike the WK estimator, the number of components will grow with time, and so the processing advantage is transient.

In figures 7 and 8 we see the single normal estimator performs as well as the kernel estimator, with the added bonus of a small PoD. This means the single normal estimator reacts quicker to the sudden change, and stabilizes in the same amount of time as the kernel estimator.

The second experiment is designed to show the performance on a distribution where the variance changes after the jump (figure 9). A physical reason for this phenomena might be the replacement of a noisy sensor, causing the new data to have smaller variance. For all methods, the performance on the smaller variance distribution is worse than on the larger variance distribution.

This experiment (in particular, figure 10) shows the main drawback of the kernel estimator approach. It is impossible to have a single bandwidth for both distributions and still obtain optimal performance. The bandwidth used is the optimal for the second distribution (given the number of points used in the kernel). The AM is less sensitive to initial conditions, and figures 10, 12, and 13 show this. Note, unlike the first experiment, no new components are created at the jump (figure 11). This is a result of our creation rule, which only creates when the data are poorly covered by the current estimate. Figure 12 shows the AM which creates every 50 points is extremely slow to stabilize, although it has the best final MISE of all.

The third experiment is designed to show the flexibility of the AM in dealing with more complicated densities. Here all the densities are mixtures, and there is considerable overlap of the densities before and after the jump (figure 14).

This experiment shows a drawback of the AM. It can be relatively slow to stabilize, in contrast to a WK, where the estimator stabilizes as soon as the window fills up with points from the new distribution. Figures 15 and 16 show this phenomena. In figures 18 and 19 we see again the AM which creates every 50 has the best MISE.

Overall, the AM can be seen to be slower to reach a steady state than the kernel estimator. Thus, although it is generally faster to react to the change than the kernel estimator, it does take longer to adapt to the change.

We now apply the estimator to real data. Cobb (1978) analyzes a data set consisting of the annual volume of discharge from the Nile River from 1871 to 1970. He assumes the data are from a jump nonstationarity, and calculates the jump to occur at 1898. Assuming the data to be normally distributed before and after the jump, we modeled the densities using the AM constrained to consist of a single Gaussian. The window width W was 10, and the starting variance was 12000. The results are shown in figures 21 and 22. The solid line is the mean of the Gaussian, and the dotted line corresponds to one standard deviation. In figure 21, the estimated parameters are plotted against the data, while in figure 22, the estimated parameters are plotted against the "true" parameters: that is the sample mean and standard deviation assuming a jump at 1898.

The AM was then applied to this data with a create threshold of 0.6. A total of six components were created, and the results are shown in figures 23 through 25. The mean and standard deviations shown in figures 23 and 24 are calculated from the density estimate. The result agrees quite well with the results of the single component case. The number of components created is plotted against time in figure 25.

Finally, to show the AM estimator is not limited to mixture densities, we look at a density that is not well modeled by a mixture of Gaussians: the uniform distribution. We simulated 1000 data points drawn from a uniform distribution and estimated the density using both an AM estimator and a kernel estimator. The AM estimator had a window W of 200, an initial variance of 0.001, and a create threshold of 0.9. It created a total of 47 components. The kernel estimator had a window of 398 (i.e., 2W–2) and a bandwidth of 0.001. Figures 26 and 27 show the estimated densities for the kernel estimator on the last 398 points, and the AM after the 1000th point. The similarity of the estimates is striking. Thus, we have evidence that the AM can produce as good an estimator as the kernel estimator on a large number of distributions. In this case, the AM uses nearly an order of magnitude fewer components, which is a considerable reduction in computation. While we can improve the kernel estimator by taking 798 points (4W–2, see equation 23), this is at a cost of doubling the computations and storage requirements of the estimator. Thus the advantage of the AM is clear.

DISCUSSION

The AM approach is a flexible technique for estimating the densities of random variables with a jump nonstationarity. We have compared the performance of this estimator with a WK estimator on a number of mixture distributions. We have also noted that the utility of the AM is not limited to mixtures.

The AM compares favorably with a WK estimator, but like everything, it has its drawbacks. Like the kernel estimator, the AM has a number of parameters that must be chosen. These are the initial variance of the components, the create threshold, and the window width. The last of these is shared by any algorithm that relies on weighting the new data more heavily than old data. The create threshold and initial variance are similar to the bandwidth of the kernel estimator in that they are chosen with the densities to be estimated in mind. A good rule of thumb, if the density does not appear to be a mixture, is to pick a small initial variance and a large create threshold, since it is generally better to have too many components than too few. Since the algorithm allows the estimator to adjust its parameters to better represent the data, the estimator tends to be less sensitive to choices of these initial parameters than the kernel estimator is to the choice of bandwidth.

As the results show, the main advantage the AM has over the kernel estimator is the ability of a single set of initial conditions to perform well on a large number of densities. This advantage eliminates the problem the kernel estimator has, which is that the optimal bandwidth for the density before the jump may be a very poor choice for the density after the jump. While in principle the kernel estimator can be modified to adjust the bandwidth to the data, this can be very expensive computationally.

Another advantage of the AM is it requires much fewer components, in general, than the kernel estimator. This makes it much more attractive for applications where data rates are high and the estimates must be computed quickly.

The AM estimator that creates every 50 points is consistently better in these experiments than either the single component AM or the one that uses the create threshold. This lends credence to the earlier claim that the more components the better. However, this performance increase comes at a price. Like the standard kernel estimator, the number of components grows without bound. Furthermore, the more components the AM uses, the slower it is to change. Thus, for densities that are not well modeled by a mixture of a few Gaussians, the kernel estimator might be a better choice, assuming the bandwidth problem can be solved.

One issue that should be considered from a practical standpoint is the deletion of components. There are two reasons to delete a component. The first reason has to do with computational considerations. Once a component's proportion has dropped to a point where it no longer contributes significantly to the estimate, it could be deleted in order to reduce the number of computations needed. Another reason is, with the creation rule used in this paper, there is no way to create a component that is "close" to an existing component, even if the proportion of the existing component is zero. Thus, one might wish to either modify the creation rule to handle this case, or allow the algorithm to delete components.

The results of this study show the AM to be a promising algorithm for density estimation in the presence of a jump nonstationarity. While the approach, like any other has its drawbacks overall, the AM and its parametric variants have shown themselves to be a powerful tool for density estimation.

REFERENCES

Carlstein, E. 1988. "Nonparametric Change-Point Estimation," *The Annals of Statistics*, 16, 188–197.

Chan, T.F., G. H. Golub, and R.J. Leveque. 1983. "Algorithms for Computing the Sample Variance: Analysis and Recommendations," *The American Statistician*, 37, 242–247.

Cobb, G. W. 1978. "The Problem of the Nile: Conditional Solution to a Change-point Problem," *Biometrika*, 65 243–251.

Silverman, B. W. 1986. "Density Estimation for Statistics and Data Analysis," London, Chapman and Hall.

Titterington, D. M. 1984. "Recursive Parameter Estimation using Incomplete Data," Journal of the Royal Statistical Society. B, 46 257–267.

Titterington, D. M., A.F.M. Smith, and U.E Makov. 1985. "Statistical Analysis of Finite Mixture Distributions," Chichester, Wiley.

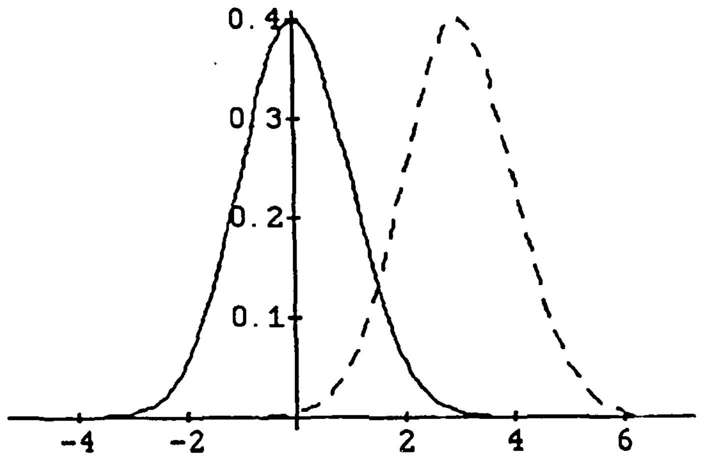

Figure 1. E1: d_1 = N(0,1), d_2 = N(3,1).

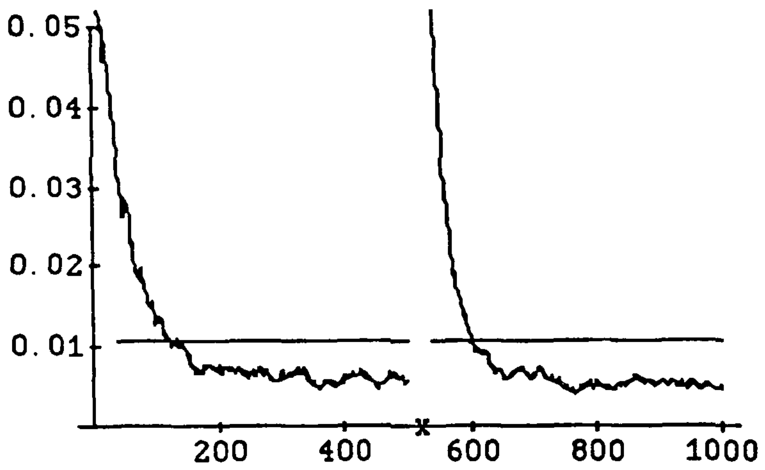

Figure 2. MISE for E1, C = 0.1, detection 19 points after the jump, kernel width = 38.

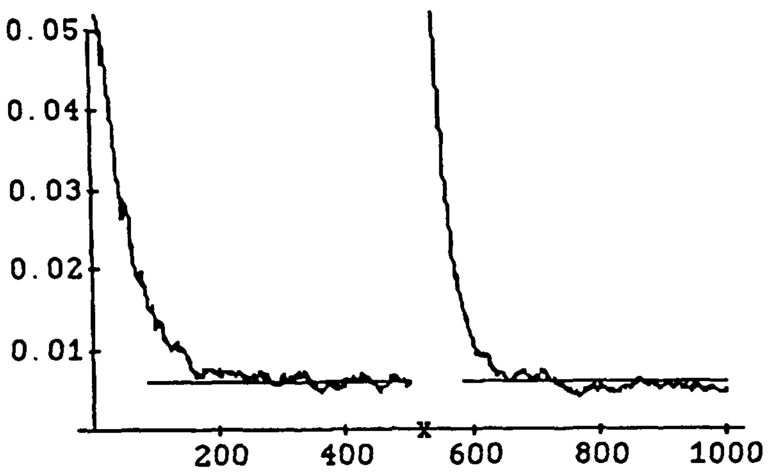

Figure 3. MISE for E1, C = 0.1, detection 19 points after the jump, kernel width = 86.

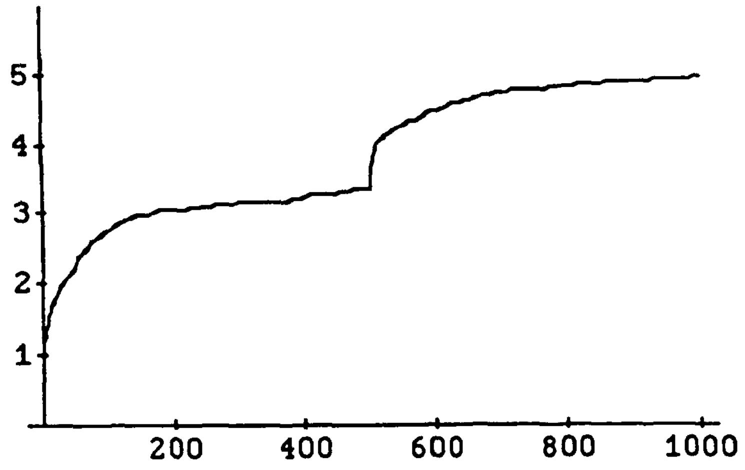

Figure 4. Number of components for E1, C = 0.1.

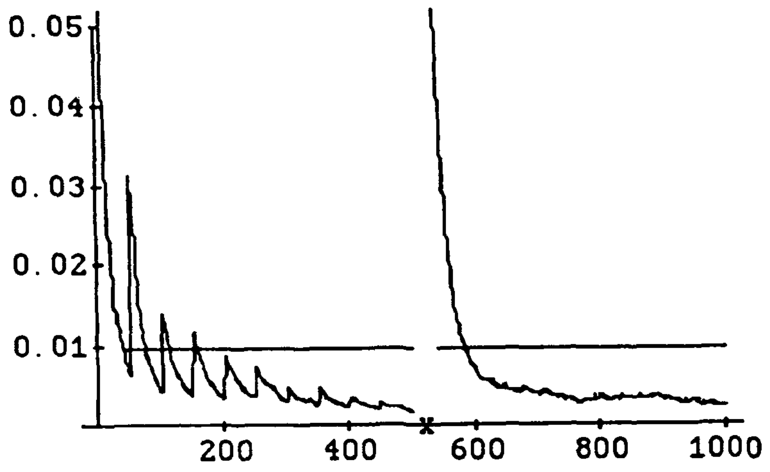

Figure 5. MISE for E1, create every 50, detection 22 points after the jump, kernel width = 44.

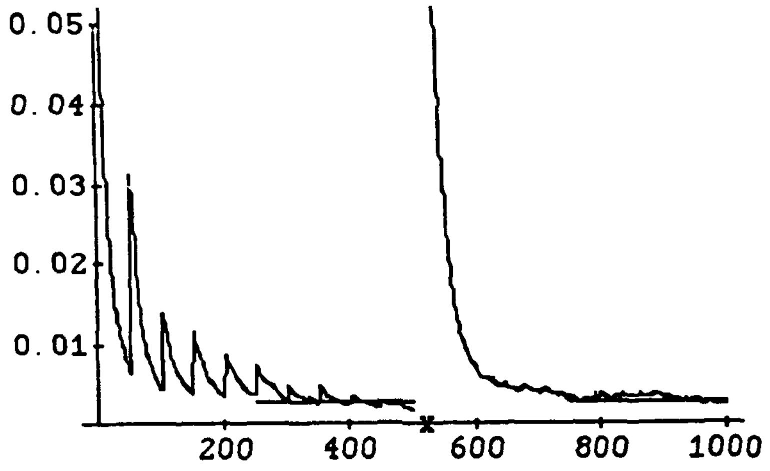

Figure 6. MISE for E1, create every 50, detection 22 points after the jump, kernel width = 250.

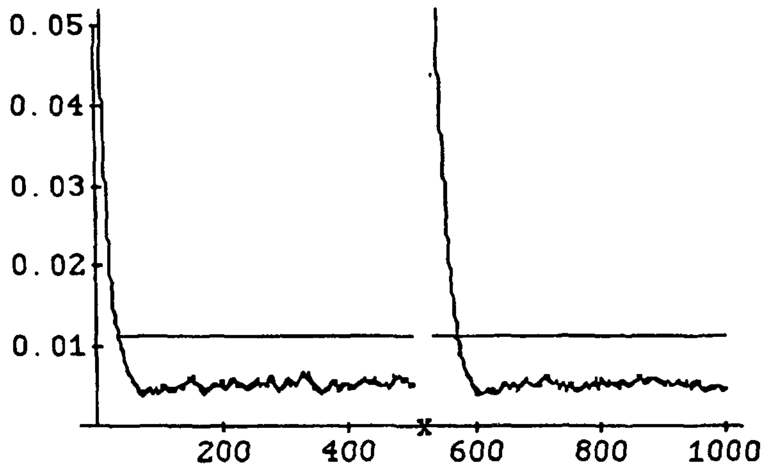

Figure 7. MISE for E1, single normal, detection 17 points after the jump, kernel width = 34.

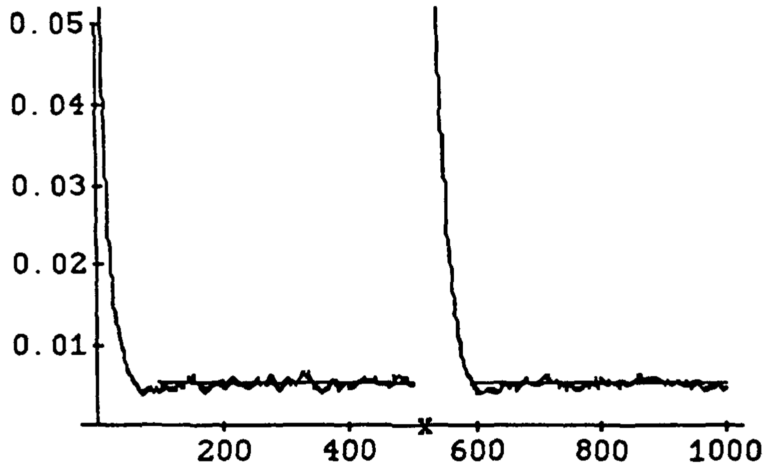

Figure 8. MISE for E1, single normal, detection 17 points after the jump, kernel width = 99.

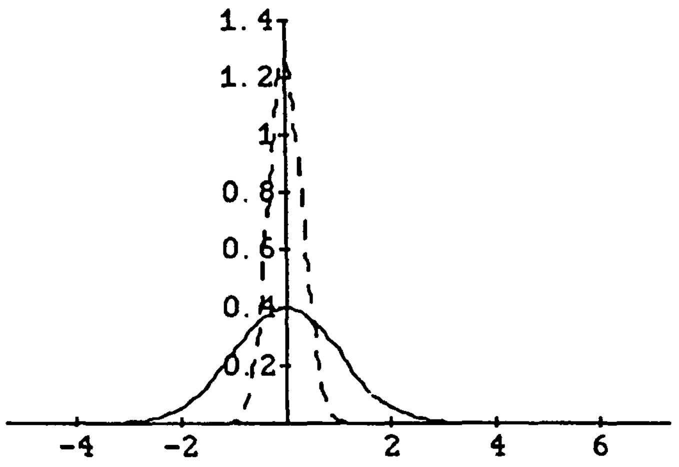

Figure 9. E2: d_1 = N(0,1), d_2 = N(0,0.1).

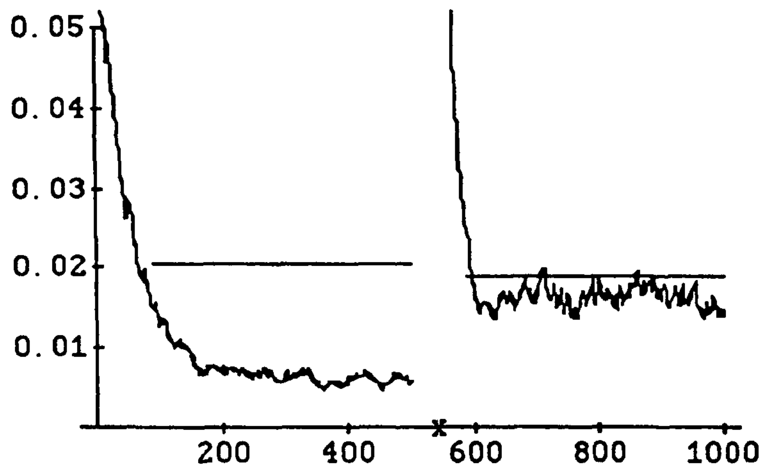

Figure 10. MISE for E2, detection 44 points after the jump, kernel width = 88.

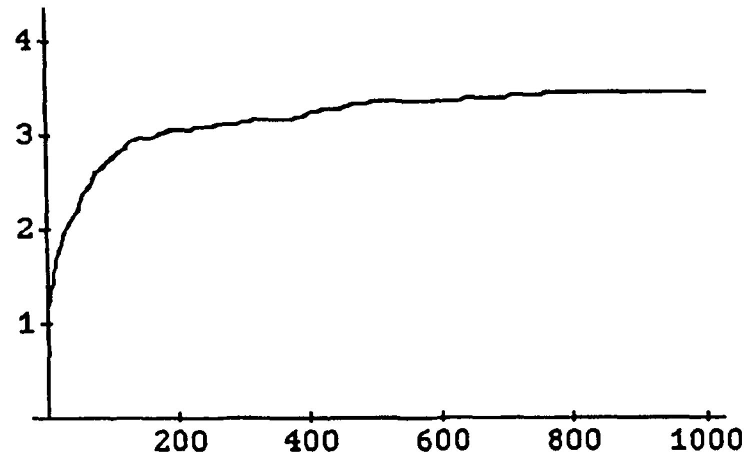

Figure 11. Number of components for E2, C = 0.1.

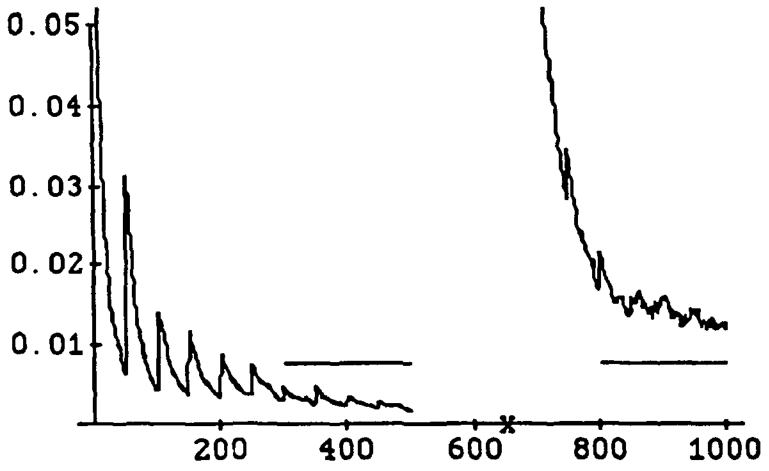

Figure 12. MISE for E2, create every 50, detection 152 points after the jump, kernel width = 302.

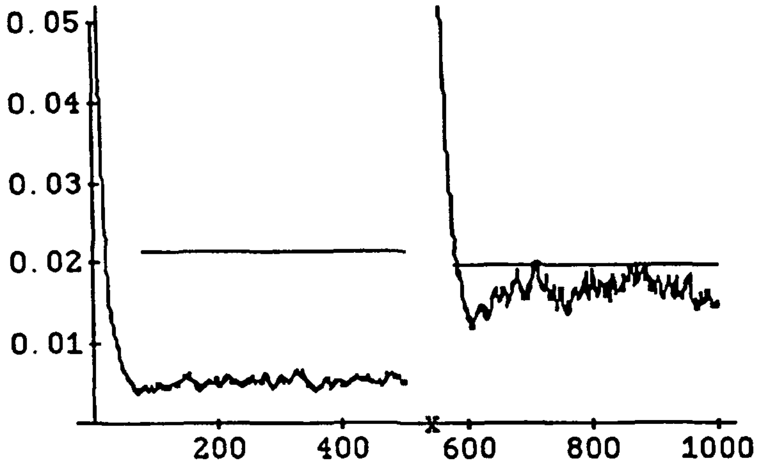

Figure 13. MISE for E2, single normal, detection 41 points after the jump, kernel width = 42.

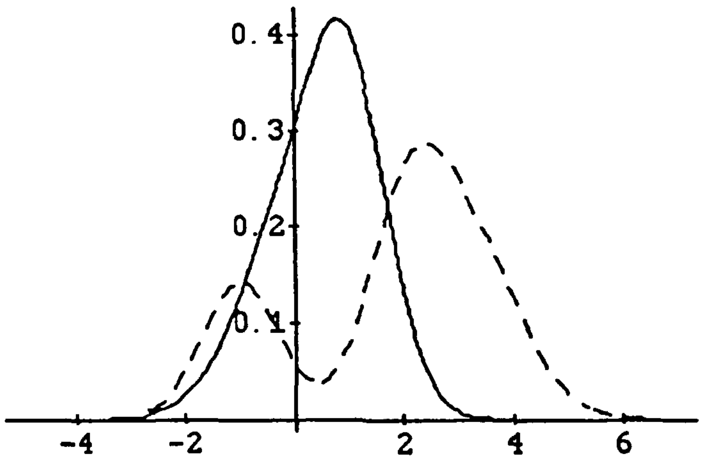

Figure 14. E3: $d_1 = \frac{1}{2}N(0,1)+\frac{1}{2}N(1,0.5)$, $d_2 = \frac{1}{2}N(3,1)+\frac{1}{4}N(2,0.5)+\frac{1}{4}N(-1,0.5)$.

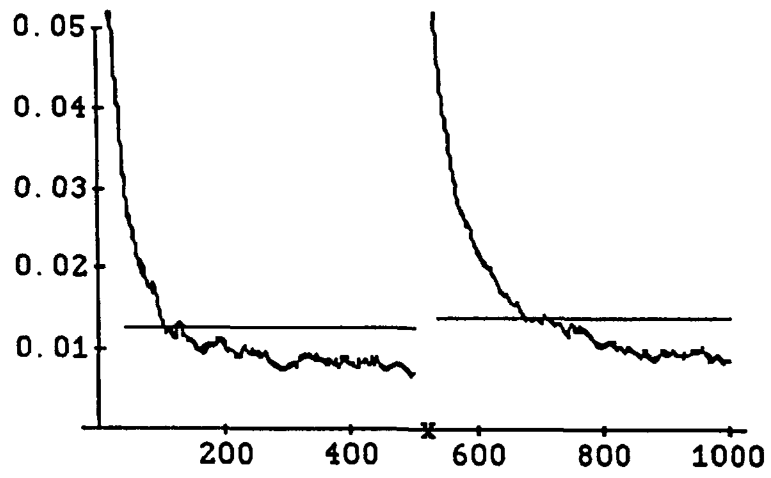

Figure 15. MISE for E3, C = 0.1, detection 20 points after the jump, kernel width = 40.

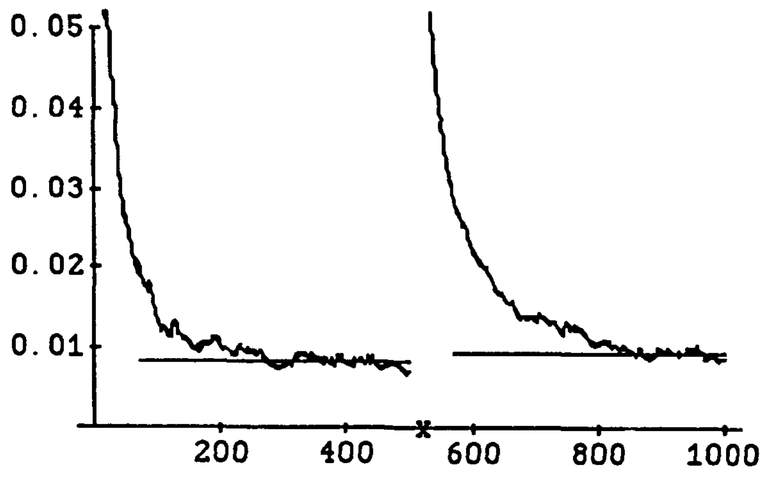

Figure 16. MISE for E3, C = 0.1, detection 20 points after the jump, kernel width = 70.

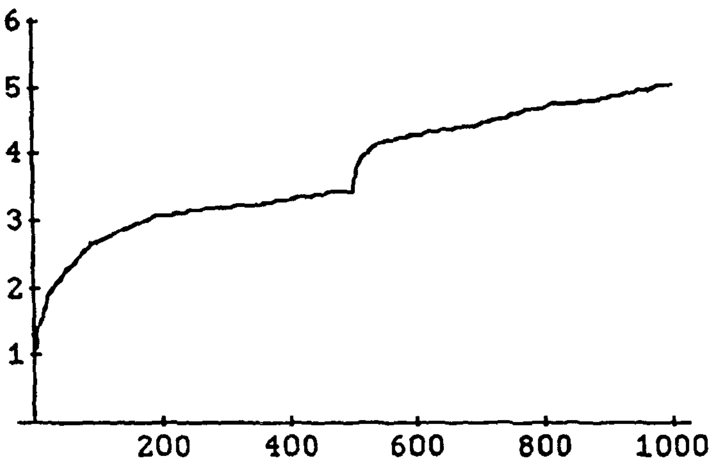

Figure 17. Number of components for E3, C = 0.1.

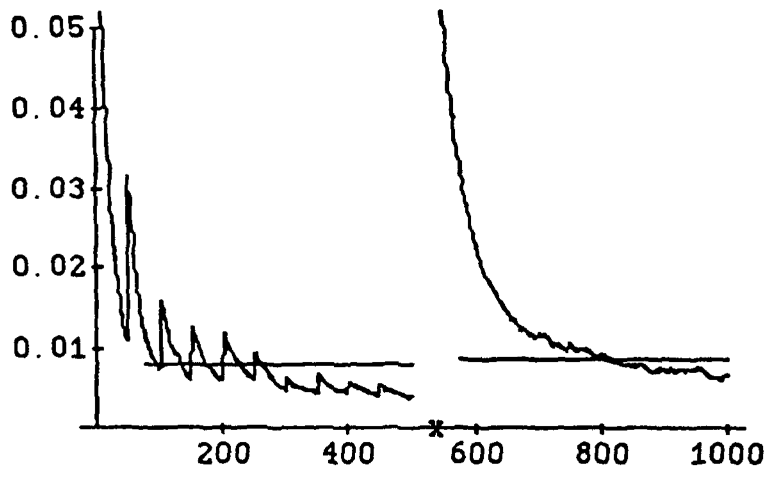

Figure 18. MISE for E3, create every 50, detection 38 points after the jump, kernel width = 76.

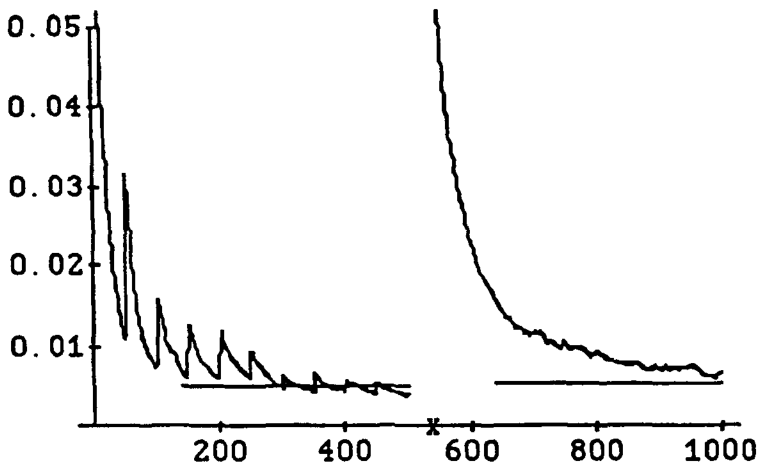

Figure 19. MISE for E3, create every 50, detection 38 points after the jump, kernel width = 140.

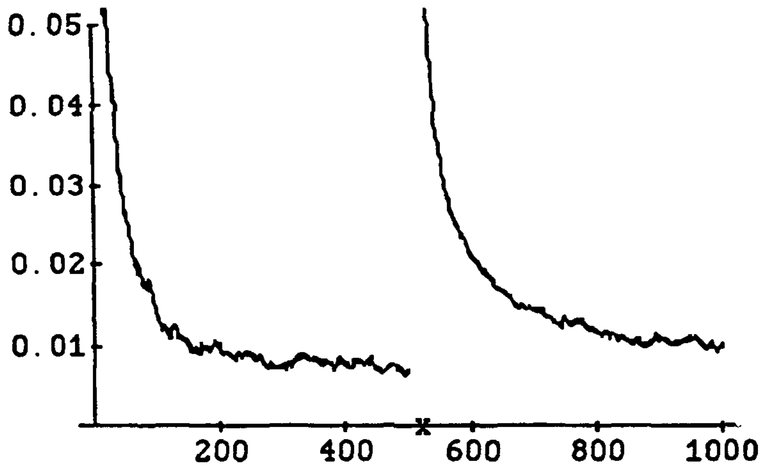

Figure 20. MISE for E3, create only 3 components, detection 22 points after the jump.

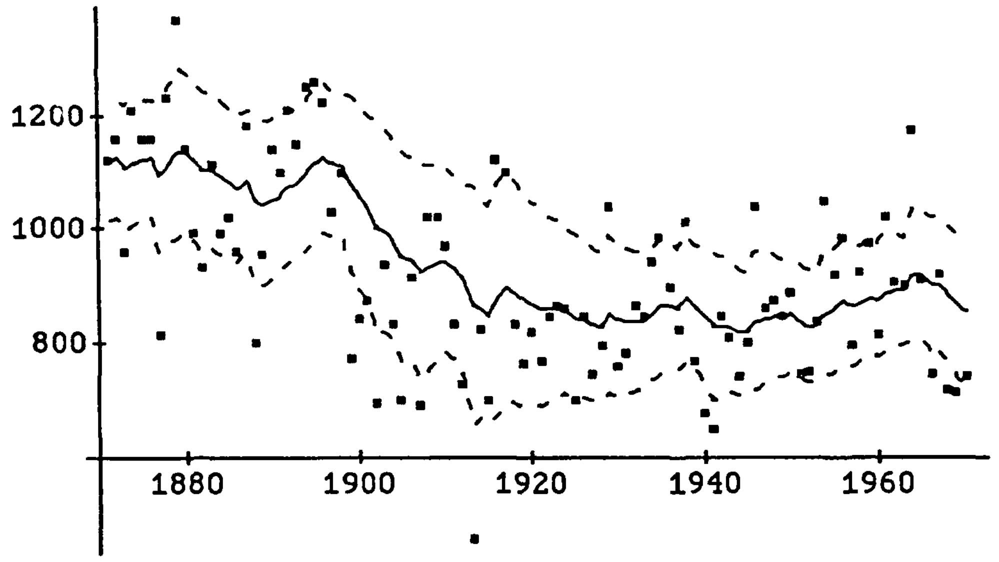

Figure 21. The Nile Data: Annual discharge 1871 to 1970.

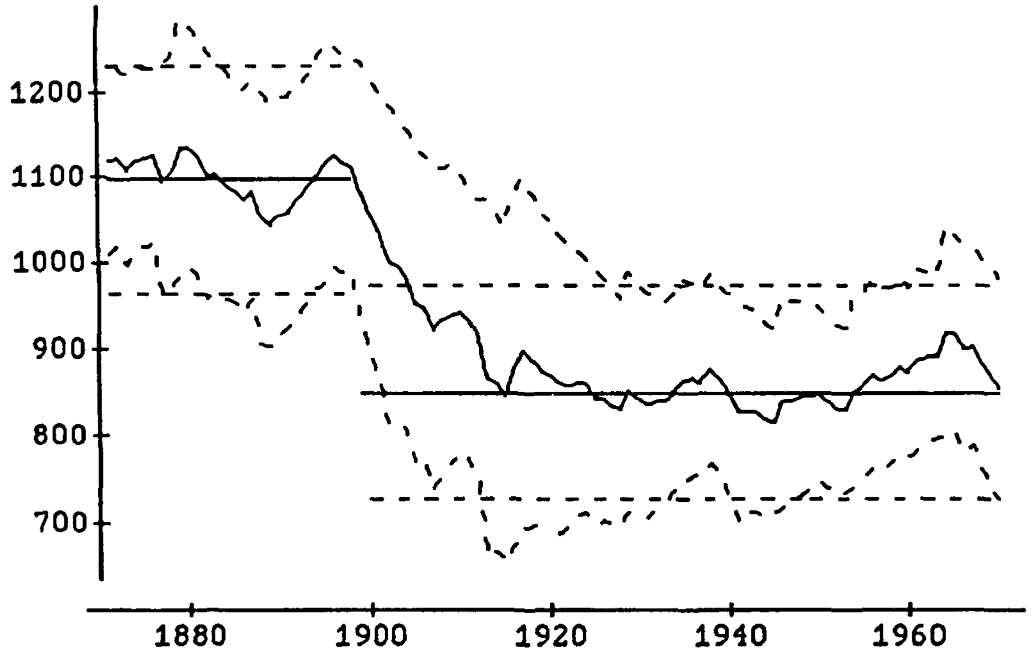

Figure 22. The Nile Data: Estimated parameters vs "true."

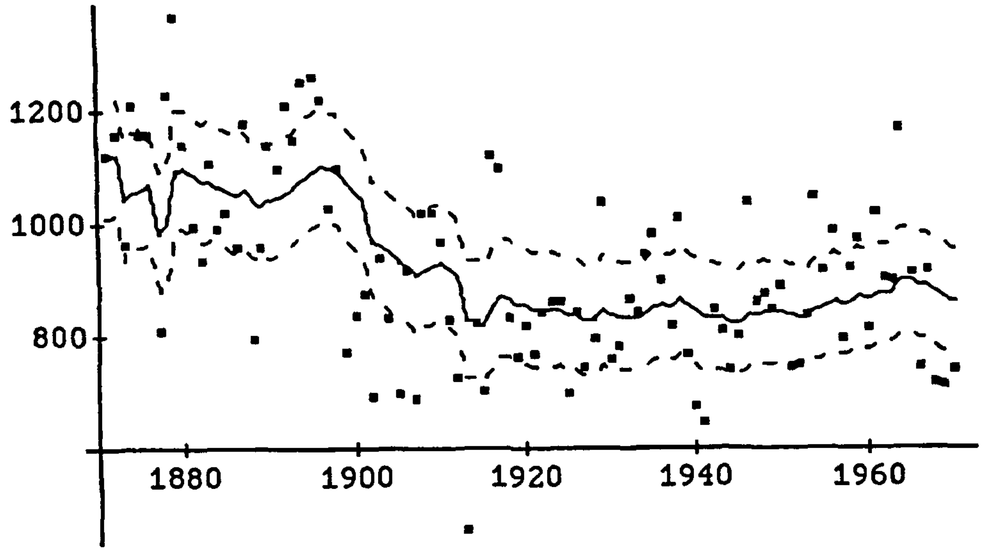

Figure 23. The Nile Data: Create threshold = 0.6.

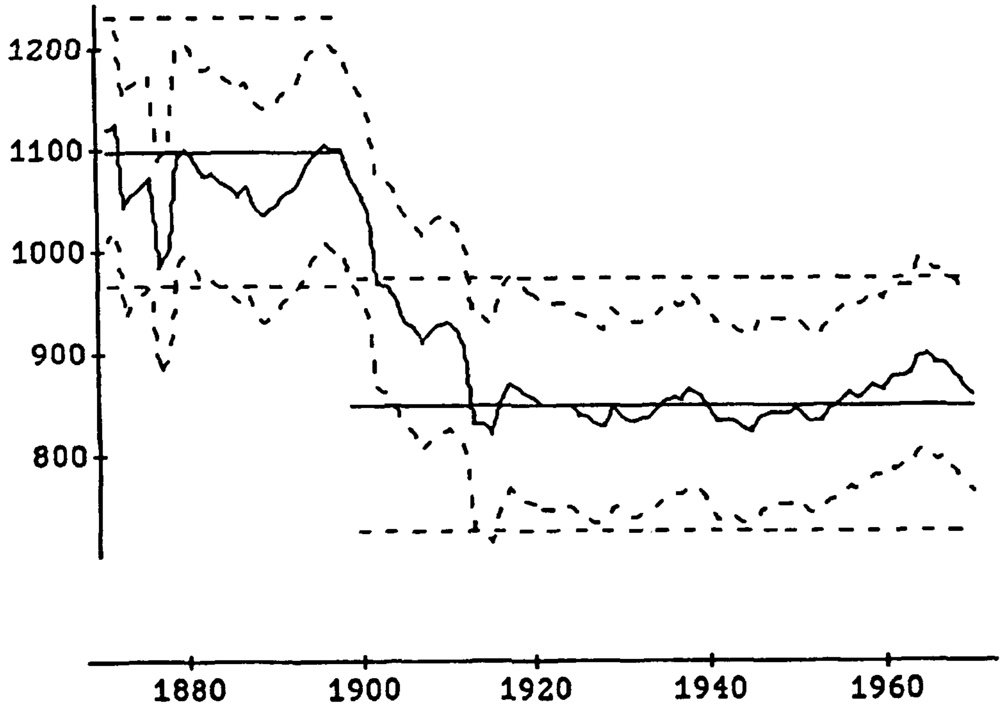

Figure 24. The Nile Data: Estimated parameters vs "true," create threshold = 0.6.

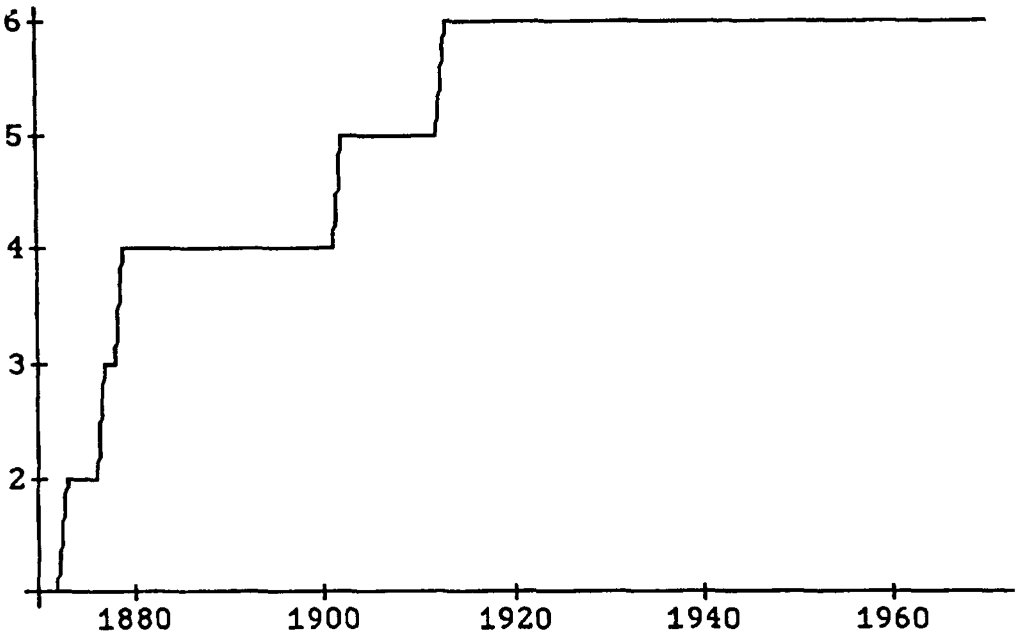

Figure 25. Number of components.

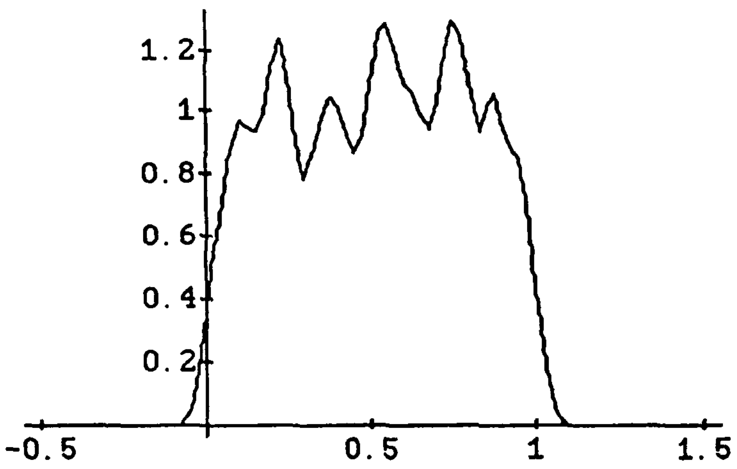

Figure 26. 398-point kernel estimator on uniformly distributed data.

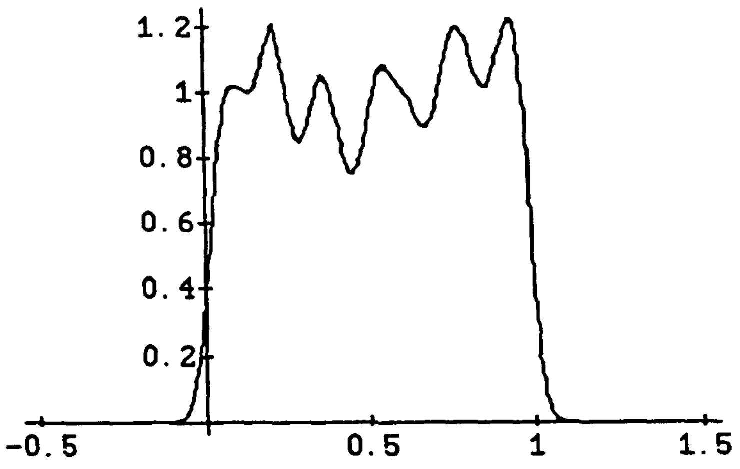

Figure 27. Adaptive mixtures approximation to uniform distribution, window width = 200, create threshold = 0.9, initial variance = 0.001.

Table 1. Experiments E1.

Estimator	MISE before	PoD	MISE after	Window	#Components
AM	0.00605	19	0.00519	25	5
Kernel 1	0.01059	19	0.01059	38	38
Kernel 2	0.00605	43	0.00605	86	86
AM50	0.00249	22	0.00285	25	20
Kernel 1	0.00959	22	0.00959	44	44
Kernel 2	0.00284	125	0.00284	250	250
Normal	0.00549	17	0.00509	25	1
Kernel 1	0.01141	17	0.01141	34	34
Kernel 2	0.00549	49	0.00549	99	99

Table 2. Experiments E2.

Estimator	MISE before	PoD	MISE after	Window	#Components
AM	0.00605	44	0.01558	25	4
Kernel 1	0.02026	44	0.01883	88	88
AM50	0.00249	152	0.01322	25	20
Kernel 1	0.00777	152	0.00781	304	304
Normal	0.00549	41	0.01610	25	1
Kernel 1	0.02139	41	0.01978	82	82

Table 3. Experiments E3.

Estimator	MISE before	PoD	MISE after	Window	#Components
AM	0.00799	20	0.00935	25	6
Kernel 1	0.01253	20	0.01400	40	40
Kernel 2	0.00836	35	0.00916	70	70
AM50	0.00468	38	0.00701	25	20
Kernel 1	0.00787	38	0.00860	76	76
Kernel 2	0.00503	70	0.00540	140	140
AM3	0.00755	22	0.01047	55	3

APPENDIX - THE ALGORITHM

```
Constants:
        create_threshold = .1;
        initial_variance = 1.;
        window = 25;

learn(x)
{
        /* Compute the value of the mixture on the input x */
        for(i = 0; i<number_components; i++){
                    $s[i] = \text{Exp}(- (x - \text{mean}[i])^2/(2\ \text{variance}[i]) );$
                    $g[i] = s[i]/ \sqrt{2\pi\ \text{variance}[i]} ;$
        }
        output = Sum( p[i] * g[i] );
        /* Decide whether to create a new component or update the old ones */
        if( Max( s[i] ) < create_threshold)
                create_component();
        else
                update_components();
  }

create_component()
{
        /* Initialize the parameters of the new component */
        means[number_components] = x;
        variance[number_components] = default_variance;
        p[number_components] = 1 / Max( number_components, 1 );
        /* Normalize the proportions to sum to 1 */
        for(i = 0; i< = number_components; i++)
                p[i] = p[i] / Sum( p[i] );
        number_components = number_components + 1;
}

update_components()

{
        for(i = O); i<number_components; i++){
                /* Compute the proportion to update the component */
                $amt = \frac{g[i] * p[i]}{output};$
                $variance[i] = variance[i] + \frac{amt}{(window + amt - 1)} *$
                    $\left(\frac{window}{window + amt} * (means[i] - x)^2 - variance[i]\right);$

                $mean[i] = mean[i] + \frac{amt}{window} * (x - mean[i]);$

                $p[i] = p[i] + \frac{amt - p[i]}{window};$
        }
}
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1 AGENCY USE ONLY *(Leave blank)*	2 REPORT DATE September 1991	3 REPORT TYPE AND DATES COVERED Final

4 TITLE AND SUBTITLE MODELING THE DENSITY OF A DISTRIBUTION CONTAINING A JUMP NONSTATIONARITY	5 FUNDING NUMBERS PR: ZW13 PE: 0601152N WU: DN309032
6 AUTHOR(S) D. J. Marchette	
7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Ocean Systems Center San Diego, CA 92152–5000	8 PERFORMING ORGANIZATION REPORT NUMBER NOSC TR 1443
9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Chief of Naval Research Independent Research Programs OCNR–10P Arlington, VA 22217–5000	10 SPONSORING/MONITORING AGENCY REPORT NUMBER
11 SUPPLEMENTARY NOTES	
12a DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b DISTRIBUTION CODE

13 ABSTRACT *(Maximum 200 words)*

An algorithm for density estimation in the presence of a jump nonstationarity is described. This approach is compared to an approach using a kernel estimator windowed to use a fixed number of data points.

Assume a random variable X is distributed with probability density function (pdf) d_1 before time t_J and with pdf d2 after time t_J. Neither the densities nor t_J are known. The data are viewed as arriving sequentially, with a requirement to report the pdf estimate after each point. A technique for density estimation under these conditions is proposed which models the density as a mixture of normal distributions. The nonstationarity of the data is accounted for through the use of a weighted window. This approach is compared to a windowed kernel estimator.

14 SUBJECT TERMS neural networks unsupervised learning pattern recognition	15 NUMBER OF PAGES 43
	16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18 SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19 SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20 LIMITATION OF ABSTRACT SAME AS REPORT

UNCLASSIFIED

21a. NAME OF RESPONSIBLE INDIVIDUAL D. J. Marchette	21b. TELEPHONE *(include Area Code)* (619) 553-4049	21c. OFFICE SYMBOL Code 421

NSN 7540-01-280-5500

Standard form 298

UNCLASSIFIED

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 0142	K. Campbell	(1)
Code 0144	R. November	(1)
Code 421	D. Marchette	(10)
Code 421	S. Conwell	(1)
Code 421	F. Kramer	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(3)

Defense Technical Information Center Alexandria, VA 22304-6145	(4)
Center for Naval Analyses Alexandria, VA 22302-0268	(1)
Navy Acquisition, Research & Development Information Center (NARDIC) Alexandria, VA 22333	(1)